Project: Building a Game

Due dates: see section 4.

1. Objectives

Through this project, you will learn how to solve a relative complex problem by using almost all problem solving methodologies learned in the class. After this project, you are expected to solve reasonably big problems effectively.

2. Project Description

First we consider a game. *In this game, a player rolls two dice. Each die has six faces. Each face contains 1, 2, 3, 4, 5 and 6 spots. After the dice have come to rest, the sum of the spots on the two top faces is calculated. If the sum is 7 or 11 on the first throw the player wins. If the sum is 2, 3, or 12 on the first throw, the player loses. If the sum is 4, 5, 6, 8, 9 or 10 on the first throw, that sum becomes the player's "point." To win, a player must continue rolling the dice until the player rolls the point value. The player loses by rolling a 7 before rolling the point.*

If a player wins one game, s/he will earn $10. Otherwise, s/he will lose $1. A player can keep playing the game until s/he chooses to quit or his/her balance becomes 0 or less.

In this project, you are required to design the pseudocode and write a C-program, using problem solving methodologies, which allows people to play the game above on a computer.

There are totally 10 players that can play the game. All players' information (name, balance, gain so far) is stored in the file `players.txt` (see below for its logical structure). When your program is started, the balance and gain of each player should be retrieved from this file. Before a player quit your program, your program should make sure the file `players.txt` contains the latest balance and gain for each player.

When started, your program will first display a menu, which contains items: 0) top up your balance, 1) play game, 2) top 5 players by balance, 3) top 5 winners by what they have won, and 4) exit the game.

When a user chooses option 0), your program should ask his/her name first and then let the user input the amount of money s/he wants to put into the balance.

When a user choose option 1), your program should ask his/her name first and then let him/her play the game. After the name is entered,

(a) the user should see the prompt:

```
Press Enter to Roll the dice.
```

(b) Once enter is pressed, the result of the dice roll is shown, for example:

```
The values of the dices are 2 and 6 whose sum is 8.
```

(c) if the game does not end at this point, the game continues according to the rules (steps (a) - (b) are repeated)

(d) when the game ends (the player looses or wins according to the rules), he/she should see who won the game and his/her current balance, for example:

```
You win the game. Your current balance is 42.
```

Now ask the user if s/he wants to play the game again. If yes, the user should be allowed to play another game *without* entering his/her name again. When s/he chooses to stop playing, your program should go back to the menu.

See the demo (section 5) for more details.

When a user chooses option 2), your program should list the name and balance of the top 5 players in terms of their balances.

When a user chooses option 3), your program should list the name and gain of the top 5 winners in terms of what they get from this game (i.e., what they win minus what they lose in *all the* previous games they had played).

When a user chooses option 4), your program should exit.

The logical structure of `players.txt`:
The information of one player occupies one line in the file with the format
one-word name, `\t`, *balance,* `\t` *gain*

For example:

```
Peter     100  90
Emma      150  0
Richard   50   10
```

Note the gain in the file is what the player has won minus what s/he has lost, in all games they have played.

3. Requirement

a. Write main pseudocode and the pseudocode for all functions needed to solve the subproblems in the main pseudocode. Your format has to follow class discussion, e.g., main pseudocode should follow the format in page 10 of L6pseudoCodeReference.pdf, and pseudocode for functions should follow the format in page 22 of L6pseudoCodeReference.pdf. You have to demonstrate your proficiency in top down approach and problem decomposition, precise specification of subproblems for the plan of main pseudocode and pseudocode of any function, the design of functions, writing pseudocode for functions through iterative refinement, use of functions to solve (sub)problems etc. Your pseudocode should be concise (please refer to numerous examples in the lecture notes).
b. You have to use an *array of structures* to store the information about every player. You will initialize the array when your program starts using information in the file `players.txt,` update the array as a player is playing the game and flush the information in the array into the file `players.txt` after a user choose to terminate your program.
c. Translate your pseudocode to a C-program.
d. Edit and test your C-program in a *bottom up* manner.
e. To mimic throwing a dice, you can use the random number generator. Study `rand()` and `srand(…)` function in page 686 of the text book or from google. Usually a random function returns a value in a large range. However, here we only want one of 6 numbers: 1, …, 6 (where each number represent a possible result of rolling a dice). You can use **%** operator.
f. Think about what loop statements or selection statements best serve your needs when you write your pseudocode.
g. For pseudocode, you can refer to examples in lecture notes or these two specific lecture notes: `L6pseudoCodeForConstructs.pdf` and `L6pseudoCodeReference.pdf` on piazza.

4. Important Dates and Deliverables

a. This project involves many components. Each of them needs about 3+ hours. So, it is a good idea to have some milestones and start the project early.
b. First deliverables and deadline
    a. A complete pseudocode for the program. (25% of the final grade of this project)
    b. You C-program which should be fully functional correctly for at least two menu items. (25%)
    c. **Submit** your files as you usually do for lab submissions before your lab session **in the week starting from Apr 11.**
    Your grade will be 0 for this part if you miss this deadline.
    d. **Demo** your work in the lab session of the **week starting from Apr 11.**

If you miss the demo, you will get at most 40% (of the final grade) for this part.

c. Second deliverables and deadline
    a. The final pseudocode (20%), and
    b. The C-program which is fully functional in a correct manner (30%).
    c. **Submit** two files for a) and b) above as you usually do for lab submissions before your lab session in the week starting from **May 2.**
    Your grade will be 0 for this part if you miss this deadline.
    d. **Demo** your work in the lab session of the **week starting from May 2.**
    If you miss the demo, you will get at most 40% (of the final grade) for this part.
d. It is a good idea to let us know *early* if you have a good reason (at the level of *medical excuse* with *evidence*) not to finish it on time. After each deadline, **NO** excuse will be accepted unless some *unexpected and unavoidable* things happen and you have *evidence* to show that.

5. Downloadable executables

Go to
`http://redwood.cs.ttu.edu/~yzhang/cs1412/project`
download windows executable `projectDemo-windows.exe` and `player.txt` to the same folder and execute the executable.